Cinvestav

# Reporte ALU con valores con signo

Alumno: Jhontan Alexander Gomez Gamboa ;
Profesor: Luis René Vela García

## 1 Introducción experimental.

En el presente reporte se presentan los módulos elaborados para poder implementar una ALU (Arithmetic Logic Unit) por sus siglas en ingles, dicho diseño digital fue diseñado para ser parametrizable con la longitud de bits de las cantidas con las que puede operar, dichos operandos deben de ser de tipo entero con signo.

## 2 Ambiente experimetal.

Quartus (Quartus Prime 17.7 Lite Edition).

ModelSim.

## 3 Programa.

En la figura 1 podemos apreciar como esta diseñado el modulo top el cual es nuestra ALU, dicho diseño instancias todos los modulos y los conecta con sus respeticas entradas y salidas, desde el testbench vamos a configurar los valores de opcode modificando el tipo de operación que realizaran los operandos, y configuraremos de igual manera los operandos desde el portaA y el portB.

La ALU cuanta con dos banderas que nos permiten conocer cuando se ha hecho un acarreo en una suma y si todos los bits son iguales a cero.

El las figuras de la 2-9 podemos ver como estan definida la lógica de los modulos que se encargaran de hacer las 8 operaciones requeridas.

El modulo flags (figura 10) se encarga de tomar las banderas de cada uno de los módulos y mostrarlas dependiendo el valor de opcode.

El modulo controlador (figura 11) se encarga de tomar todas las salidas de las operaciones y mostrar el resultado solicitado dependiendo del valor de opcode, ademas de poder tratar el tamaño de los resultados para que compartan un mismo bufer.

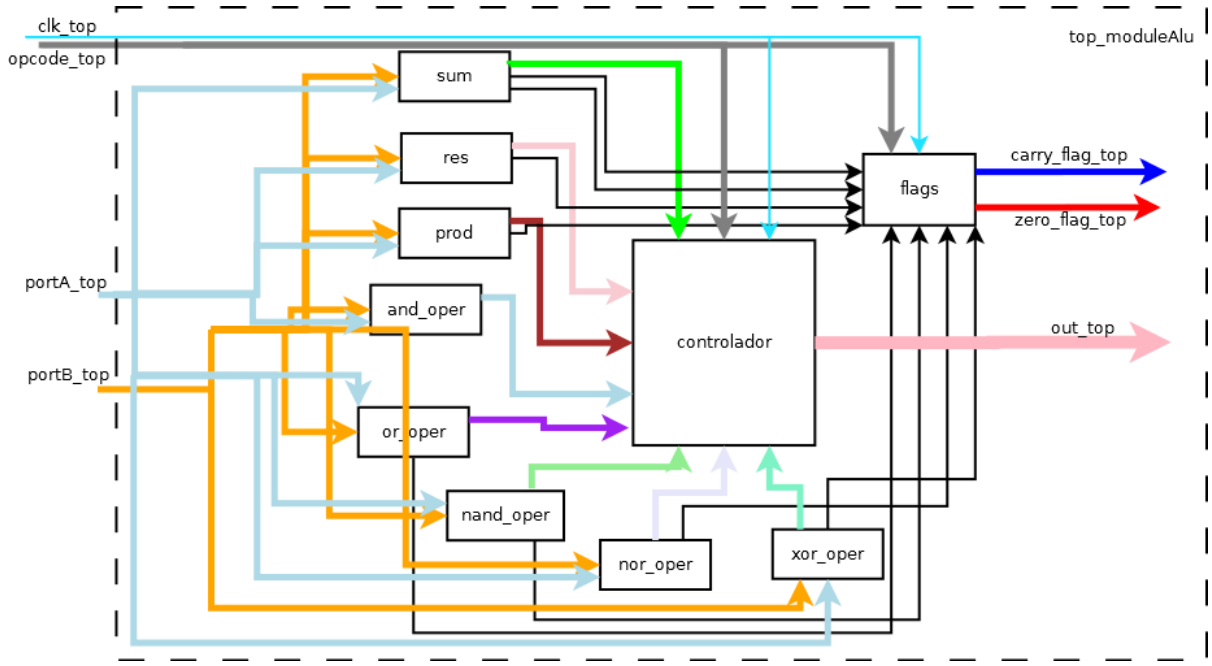


Figure 1: *Diagrama del* ***top_moduleAlu*** *el cual se encarga de isntancias cada uno de los diseñoes e interconectarlos con sus respeticas señales*

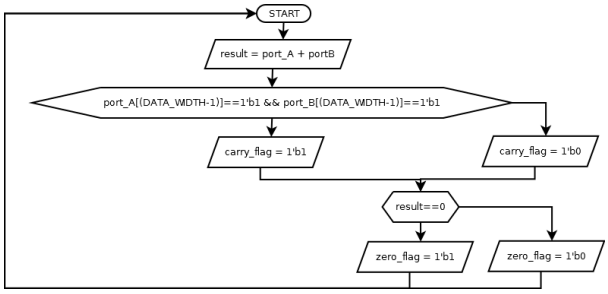


Figure 2: *Diagrama de flujo que describe el funcionamiento del módulo* ***sum***

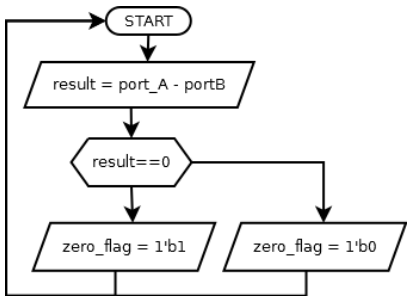


Figure 3: *Diagrama de flujo que describe el funcionamiento del módulo* ***rest***

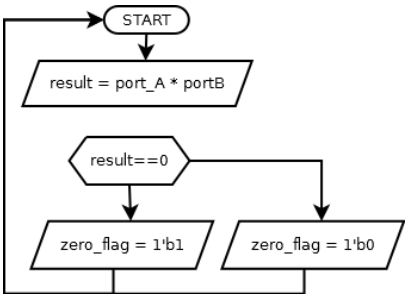


Figure 4: *Diagrama de flujo que describe el funcionamiento del módulo* ***prod***

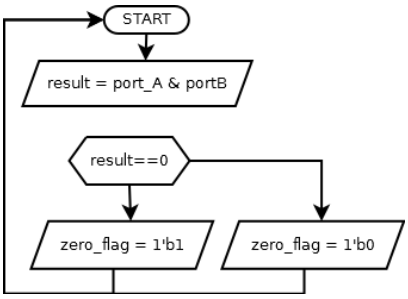


Figure 5: *Diagrama de flujo que describe el funcionamiento del módulo* ***and_oper***

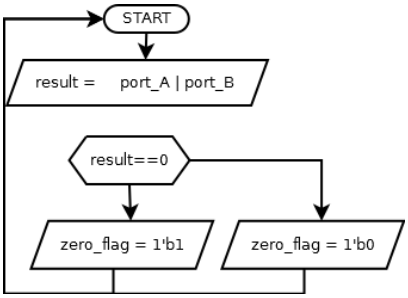


Figure 6: *Diagrama de flujo que describe el funcionamiento del módulo* ***or_oper***

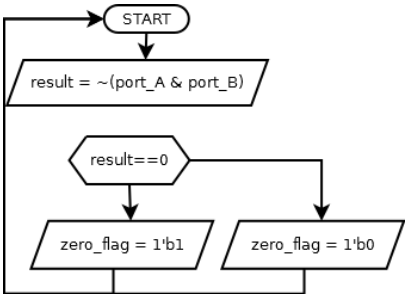


Figure 7: *Diagrama de flujo que describe el funcionamiento del módulo* ***nand_oper***

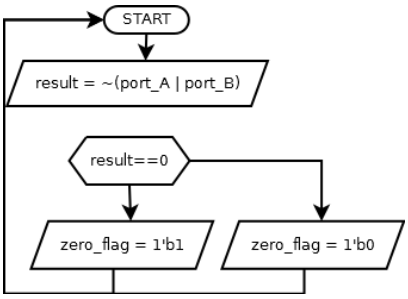


Figure 8: *Diagrama de flujo que describe el funcionamiento del módulo* ***nor_oper***

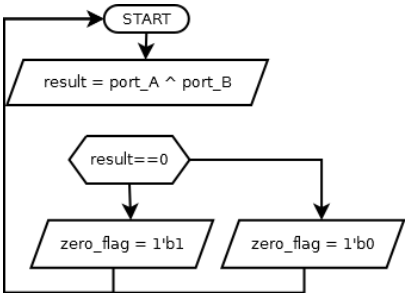


Figure 9: *Diagrama de flujo que describe el funcionamiento del módulo* ***xor_oper***

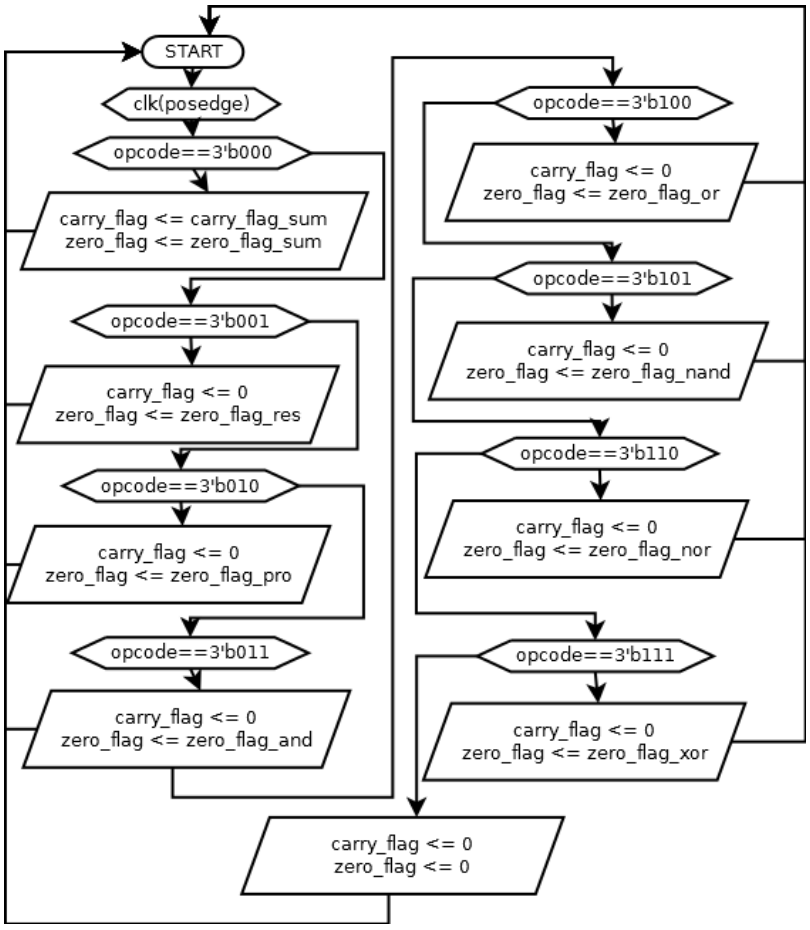


Figure 10: *Diagrama de flujo que describe el funcionamiento del módulo* ***flags***

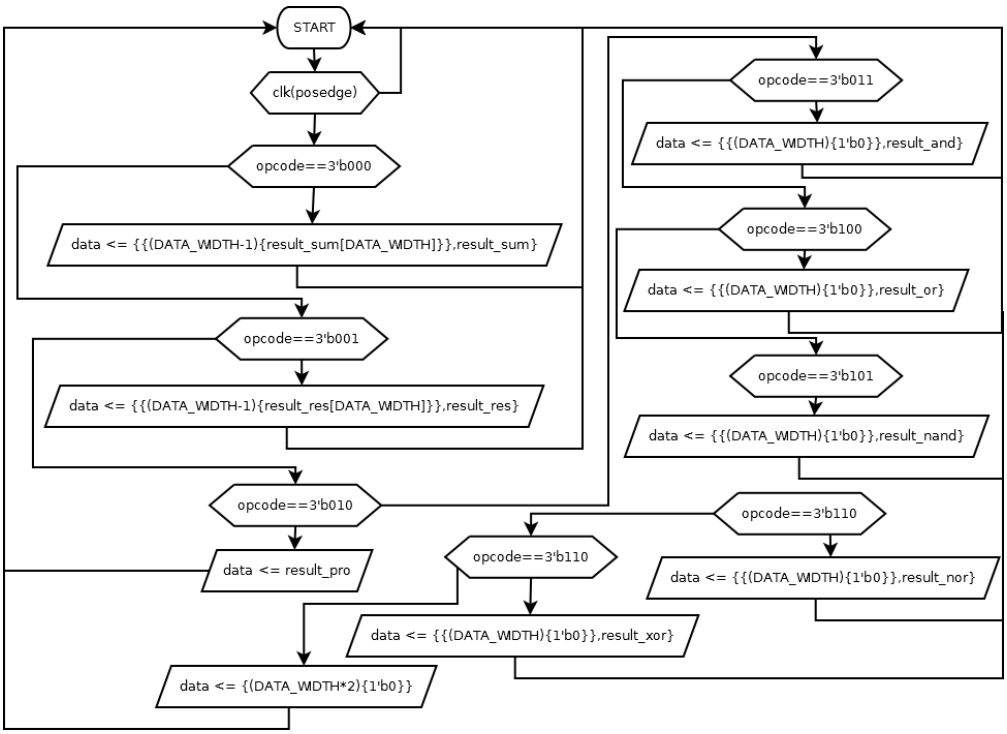


Figure 11: *Diagrama de flujo que describe el funcionamiento del módulo* ***controlador***

# 4 Resultados simulados.

En las siguiente figuras desde la 12 - 15 podemos ver las salidas con diferentes valores, tratando de tener diferentes escenarios para poder observar el correcto funcionamiento de la ALu. los siguientes son los escenarios a tratar: 1. Operador 1 (positivo) , operador 2 (positivo) 2. Operador 1 (positivo) , operador 2 (negativo) 3. Operador 1 (negativo) , operador 2 (negativo) 4. Operador 1 (negativo) , operador 2 (negativo) misma magnitud pero diferente signo.

En la siguiente tabla mostramos las operaciones con sus respectivos resultados en formato decimal para poder corroborar que la ALU es funcional con las especificaciones.

Table 1: valores de los operandos y sus resultados para el caso valor 1 =12, valor 2 =16.

| **suma** | **resta** | **producto** | **and** | **or** | **xor** |
|---|---|---|---|---|---|
| 28 | -4 | 192 | 0 | 28 | 28 |

Table 2: valores de los operandos y sus resultados para el caso valor 1 =32, valor 2 =-49.

| **suma** | **resta** | **producto** | **and** | **or** | **xor** |
|---|---|---|---|---|---|
| -17 | 81 | -1568 | 0 | 239 | 239 |

Table 3: valores de los operandos y sus resultados para el caso valor 1 =-127, valor 2 =-74.

| suma | resta | producto | and | or | xor |
|---|---|---|---|---|---|
| -201 | -53 | 9398 | 128 | 183 | 55 |

Table 4: valores de los operandos y sus resultados para el caso valor 1 =127, valor 2 =-127.

| suma | resta | producto | and | or | xor |
|---|---|---|---|---|---|
| 0 | 254 | -16129 | 1 | 254 | 254 |

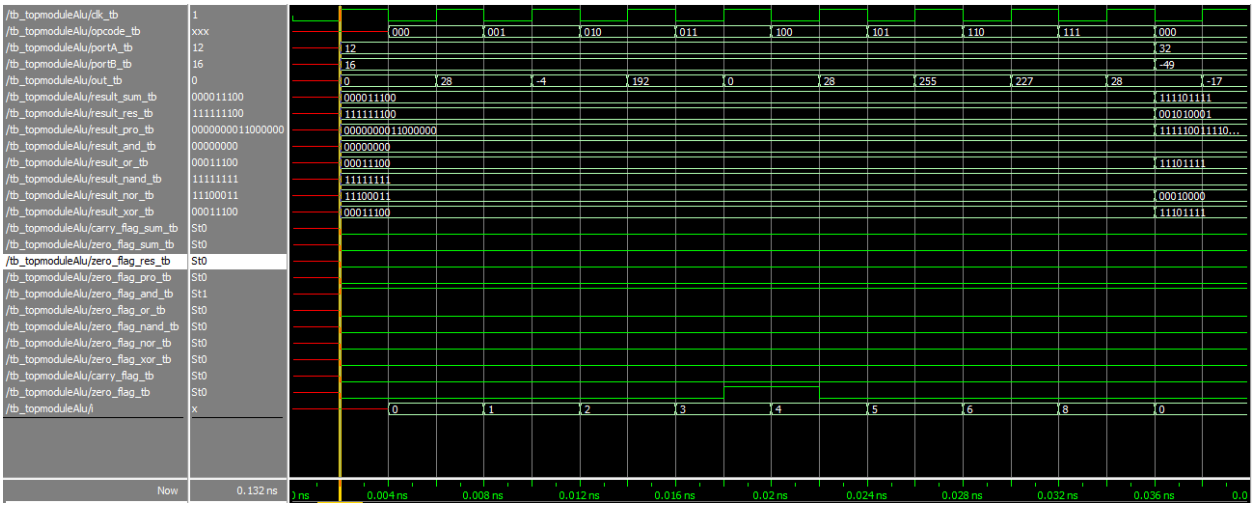


Figure 12: *imagen de la simulación donde se introducen dos operandos de 8 bits positivos 12 y 16 respectivamente, con sus respectivas salidas*

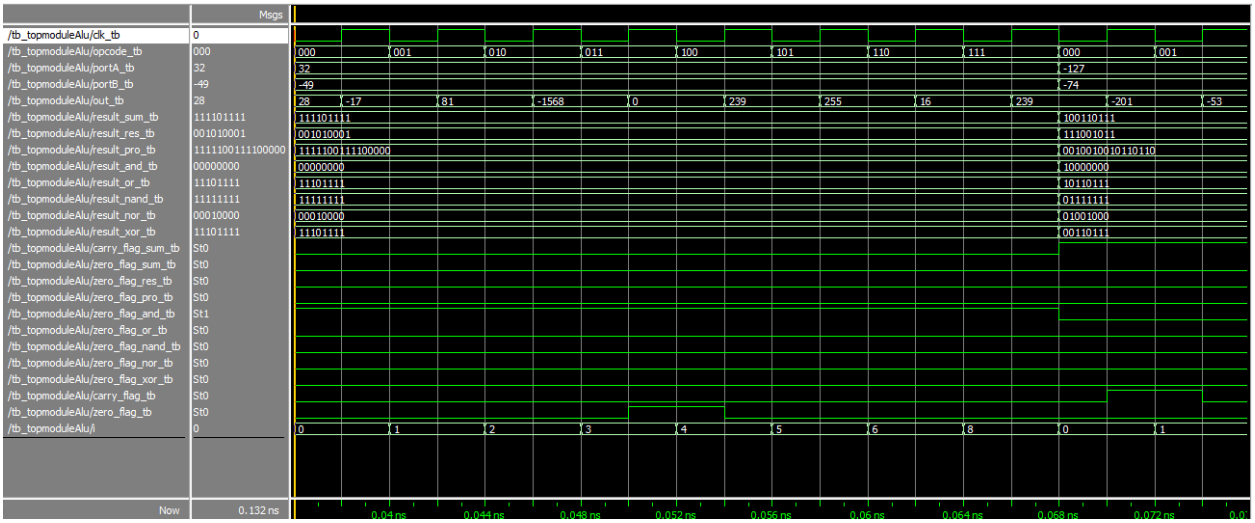


Figure 13: *imagen de la simulación donde se introducen dos operandos de 8 bits uno positivo y uno negativo 32 y -49 respectivamente, con sus respectivas salidas*

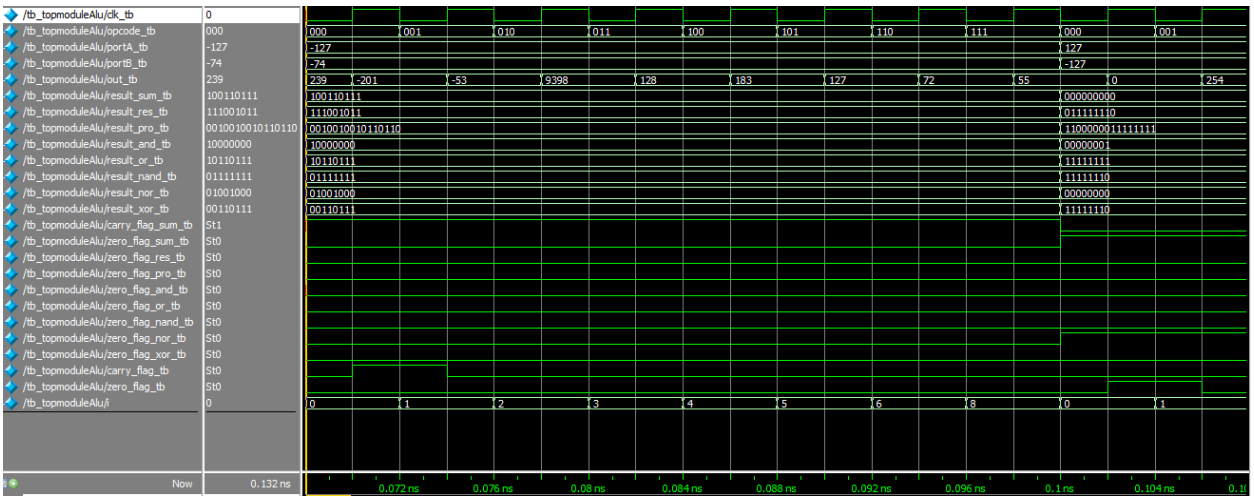


Figure 14: *imagen de la simulación donde se introducen dos operandos de 8 bits negativos -127 y -74 respectivamente, con sus respectivas salidas*

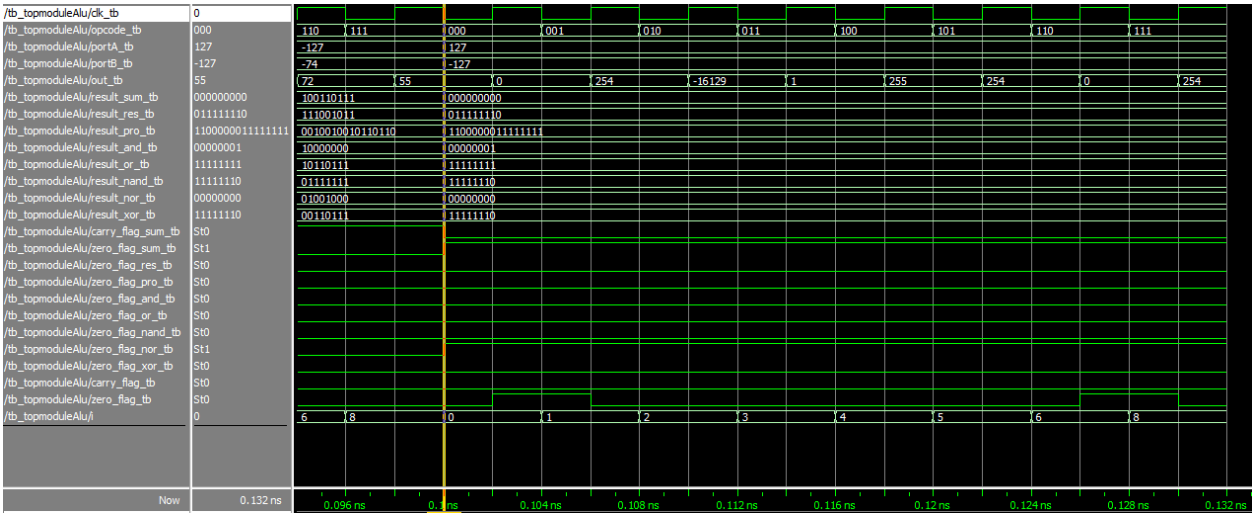


Figure 15: *imagen de la simulación donde se introducen dos operandos de 8 bits iguales con signo contrario 127 y -127 respectivamente, con sus respectivas salidas*

# 5 Valores de la cama de pruebas y respuesta a las preguntas.

```
initial begin
    clk_tb = 1'b0;
    #2;
    portA_tb = 8'b0000_1100; //12
    portB_tb = 8'b0001_0000; //16
    #34;
    portA_tb = 8'b0010_0000; //32
    portB_tb = 8'b1100_1111; //-49
    #32;
    portA_tb = 8'b1000_0001; //-127
    portB_tb = 8'b1011_0110; //-74
    #32;
    portA_tb = 8'b0111_1111; //127
    portB_tb = 8'b1000_0001; //-127
    #32;
    $stop;

end
```

Respondiendo a las preguntas planteadas en el documento del proyecto:

*Si consideramos que las entradas de la ALU son de 8 bits ¿Cuántos bits se requieren a la salida para expresar el resultado de la suma y la resta?* R : 9 bits

*¿Cuántos bits se requieren para expresar el resultado de la multiplicación?* R: 16 bits

*¿Cuántos bits se requiere expresar el bus de salida del resultado para poder expresar los resultados de todas las operaciones?* : 16 bits, el tamaño del bufer mas grande y acompletamos los lugares vacios de los datos que lo necesiten.